# LOGIC CIRCUIT AND METHOD FOR CARRY AND SUM GENERATION AND METHOD OF DESIGNING SUCH A LOGIC CIRCUIT

This application claims priority under 35 U.S.C. 119(e) from U.S. Provisional Application Serial No. 60/436,179 filed December 23, 2002, the specification of which is incorporated herein by reference and made a part hereof.

The present invention relates to a method and apparatus for use in logic circuits, and in particular, to a method and apparatus for generating a carry or sum bit by combining two binary inputs.

Addition of two binary numbers is a fundamental operation used in many electronic circuits. For example, binary addition is used in integer arithmetic-logic units, and also, all the floating-point operations use integer addition in their calculations. Memory accesses require integer addition for address generation, branches use addition for forming instruction addresses, and for making greater-than or less-than comparisons. Thus, many modern circuits contain several integer adders, many of which may appear on frequency-limiting paths.

In an addition of two numbers, the digit in each column of the first number is added to the digit in the corresponding column of the second number, and any carry digit resulting from the previous column is also added, in order to obtain the value of the sum in each column. Thus, for two n-bit binary numbers $a = a_{n-1}\ldots a_1a_0$ and $b = b_{n-1}\ldots b_1b_0$, their sum is the n+1 bit number given by $s = s_n\ldots s_1s_0$, where:

$$s_n = c_n$$

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_ib_i + c_i(a_i + b_i)$$

where $c_k$ is the carry into position k, + denotes logical OR, proximity denotes Logical AND and $\oplus$ denotes Exclusive OR.

The carry bit into any chosen column can be generated from two logical functions called Generate and Propagate. The bit level Generate function $g_i$ indicates whether a carry is generated by a particular column in the addition. The function $g_i$ is true if a carry is generated at column i. The bit-level propagate function $p_i$ indicates whether any carry for a particular column will be propagated on to the next column. The function $p_i$ is true if carry into column i is propagated into column i+1. The bit level generate and propagate functions can be constructed from the bits in column i of the two numbers to be added, as follows:

$$g_i = a_i b_i$$

$$p_i = a_i + b_i$$

Thus, in the addition of $a = a_{n-1}\ldots a_1 a_0$ and $b = b_{n-1}\ldots b_1 b_0$, the carry into the j+1'th column is given by:

$$G_{j:0} = c_{j+1} = g_j + p_j g_{j-1} + p_j p_{j-1} g_{j-2} + \ldots + p_j p_{j-1}\ldots p_1 g_0$$

Figure 1 shows an implementation of a circuit to generate $G_{j:0}$ based on the above equation. However, the circuit of figure 1 is not a practical circuit to realize for large values of j. It is an OR of j+1 AND-terms, the largest of the AND gates also having j+1 inputs. Moreover, the fan-out of the p's is very large, $p_j$ having a fan-out of j.

High speed practical implementations realize the carry function in a tree like structure. A prior art method is known as parallel prefix and will now be illustrated (S Knowles, "A Family of Adders", Proc, 14th IEEE Symp. On Computer Arithmetic, pp30-44, 1999). The parallel prefix method uses bit-level generate and propagate functions to construct Group Generate and Group Propagate functions.

$G_{j:k} = g_j + p_j g_{j-1} + p_j p_{j-1} g_{j-2} + \ldots + p_j p_{j-1} \ldots p_{k+1} g_k$

$P_{j:k} = p_j p_{j-1} \ldots p_{k+1} p_k$

The function $G_{j:k}$ is true if the group of bits from k to j generates a carry and the function $P_{j:k}$ is true if the group of bits from k to j propagates a carry coming into that group into the next group.

The parallel prefix method uses Group Generate and Group Propagate functions of smaller sized groups to construct the Group Generate and Group Propagate functions of a larger group. A large group of bits from i to j is divided into 2 groups say from i to k-1 and k to j. The larger group generates a carry if either the most significant group generates a carry or the least significant group generates a carry and the most significant group propagates this carry. This is illustrated in figure 2. In logical notation this can be expressed as.

$G_{j:0} = G_{j:k} + P_{j:k} G_{k-1:0}$

The Group Propagate function of a large group can be constructed from Group Propagate functions of smaller groups:

$P_{j:i} = P_{j:k} P_{k-1:i}$

These two constructions allow the Group Generate of a larger group to be formed recursively from smaller groups, which themselves are formed from even smaller groups and so on.

This method allows for the construction of $G_{j:i}$ in $\lceil \log_2(j - i) \rceil$ levels, once the bit-level generate and propagate functions have been formed.

It is possible to form the Group Generate of a large group in fewer levels still. If the large group i to j is divided into 3 groups say, i to k'-1, k' to k''-1, and k'' to j then:

$$G_{j:i} = G_{j:k''} + P_{j:k''} G_{k''-1:k'} + P_{j:k''} P_{k''-1:k'} G_{k'-1:i}$$

The drawback of this method is that although fewer combining levels are needed, the gates at each combining level are more complex and the fan-out on the Group Generate and Group Propagate functions increases. Both of these impact heavily on the delay of the circuit. This situation is further exasperated when all the carries for an adder need to be constructed.

The following is an example of the parallel prefix method for a 9-bit addition, using base 3. A circuit diagram for this example is shown in figure 3.

Given two 9-bit numbers $a = a_8a_7\ldots a_1a_0$ and $b = b_8b_7\ldots b_1b_0$, we form 3-bit groups $a_8a_7a_6$, $a_5a_4a_3$, $a_2a_1a_0$ for a and $b_8b_7b_6$, $b_5b_4b_3$, $b_2b_1b_0$ for b.

Then the generate and propagate functions for each group are

$$G_{8:6} = g_8 + p_8g_7 + p_8p_7g_6,\ P_{8:6} = p_8p_7p_6$$
$$G_{5:3} = g_5 + p_5g_4 + p_5p_4g_3,\ P_{5:3} = p_5p_4p_3$$
$$G_{2:0} = g_2 + p_2g_1 + p_2p_1g_0,\ P_{2:0} = p_2p_1p_0$$

These Group functions are now combined to form:

$$G_{8:0} = G_{8:6} + P_{8:6}G_{5:3} + P_{8:6}P_{5:3}G_{2:0}$$

The other carries could be constructed in the following manner:

$$G_{7:0} = G_{7:6} + P_{7:6}G_{5:3} + P_{7:6}P_{5:3}G_{2:0}$$
$$G_{6:0} = G_{6:6} + P_{6:6}G_{5:3} + P_{6:6}P_{5:3}G_{2:0}$$
$$G_{5:0} = G_{5:3} + P_{5:3}G_{2:0}$$
$$G_{5:0} = G_{5:3} + P_{5:3}G_{2:0}$$

$G_{4:0} = G_{4:3} + P_{4:3}G_{2:0}$

$G_{2:0} = g_2 + p_2g_1 + p_2p_1g_0$

$G_{1:0} = g_1 + p_1g_0$

$G_{0:0} = g_0$

An improved prior art technique for determining the carry bits is the Ling method (H. Ling, "High Speed Binary Adder", IBM Journal of Research and Development, Vol 25, No 3, pp156-166, 1981). Ling observed a variation of the above, which allows for a small speed up on the parallel prefix method. He observed that if the delay of the carry term $G_{j:i}$ could be reduced by increasing the delay of some other term, the overall delay will be reduced as long as the carry term is still on the critical path. Ling observed that every term in

$$G_{j:i} = g_j + p_jg_{j-1} + p_jp_{j-1}g_{j-2} + \ldots + p_jp_{j-1}\ldots p_{i+1}g_i$$

contains $p_j$ except for the very first term, which is simply $g_j$. However, $G_{j:i}$ can still be simplified by noting that

$$g_k = p_kg_k$$

Therefore $p_j$ can be factored out of $G_{j:i}$ to create a pseudocarry $H_{j:i}$, where

$$G_{j:i} = p_jH_{j:i}$$

$$H_{j:i} = g_j + G_{j-1:i}$$

The function $H_{j:i}$ is a little simpler than the function $G_{j:i}$. The fan-in of the OR gate for $H_{j:i}$ and $G_{j:i}$ is the same but the fan-in of each AND-gate is reduced by 1. This is illustrated in Figure 4. Ling also observed that the pseudocarry $H_{j:i}$ of a large group could be constructed from the pseudocarries $H_{j:k}$ and $H_{k-1:i}$ of smaller groups:

$$H_{j:i} = g_j + G_{j-1:i} = g_j + G_{j-1:k} + P_{j-1:k}\,G_{k-1:I}$$

$$= [g_j + G_{j-1:k}] + P_{j-1:k}\, p_{k-1}[g_{k-1} + G_{k-2:i}]$$
$$= H_{j:k} + P_{j-1:k-1}\, H_{k-1:i}$$

This provides a method for constructing the pseudocarry of a large group in terms of pseudocarries of smaller groups, which can be constructed from the pseudocarries of yet still smaller groups.

As in the parallel prefix case more than two pseudocarries can be combined to form the pseudo carry of a large group:

If the large group i to j is divided into 3 groups say, i to k'-1, k' to k''-1, and k'' to j then:

$G_{j:i} = p_j H_{j:i}$

$H_{j:i} = g_j + G_{j-1:i}$

$G_{j-1:i} = G_{j-1:k''} + P_{j-1:k''}\, G_{k''-1:k'} + P_{j-1:k''}\, P_{k''-1:k'}\, G_{k'-1:i}$

$G_{k''-1:k'} = p_{k''-1} H_{k''-1:k'}$

$G_{k'-1:i} = p_{k'-1} H_{k'-1:i}$

$H_{j:i} = H_{j:k''} + P_{j-1:k''-1} H_{k''-1:k'} + P_{j-1:k''-1} P_{k''-2:k'-1} H_{k'-1:i}$

This method still suffers the same problems as the parallel prefix method, that is, more complex gates. Note that $H_{j:i}$ has the form $H_2 + P_2H_1 + P_2P_1H_0$, which is exactly the same as that of the Group generate function $G_2 + P_2G_1 + P_2P_1G_0$ in the parallel prefix method, and higher fan-out is the also the same. Ling's method will now be illustrated by way of example.

The following is an example of a 9-bit Ling adder, which is illustrated in figure 5a.

$G_{8:0} = G_{8:6} + P_{8:6}G_{5:3} + P_{8:6}P_{5:3}G_{2:0} = p_8\, H_{8:0}$

$H_{8:0} = H_{8:6} + P_{7:5}H_{5:3} + P_{7:5}P_{4:2}H_{2:0}$

The pseudocarry functions are:

$H_{8:6} = g_8 + g_7 + p_7g_6$

$H_{5:3} = g_5 + g_4 + p_4g_3$

$H_{2:0} = g_2 + g_1 + p_1g_0$

Note that at the first level, the highest complexity function for Ling has the form $H_2 + H_1 + P_1H_0$, where as for parallel prefix this is $G_2 + P_2G_1 + P_2P_1G_0$.

But the complexity of $H_{8:0}$ is the same as $G_{8:0}$, both being of the form A + BC + DEF. One may try to combine $P_{7:5}$ $P_{4:2}$ and thus reduce the complexity of the second level to A + BC + DE, but

$P_{7:5}$ $P_{4:2} = P_{7:2} = p_7$ $p_6$ $p_5$ $p_4$ $p_3$ $p_2$

which is an AND of 6 terms and generally slower to calculate than

$H_{2:0} = g_2 + g_1 + p_1g_0$

The Ling adder does have the problem that to produce the actual carry out the logical AND of $p_j$ and $H_{j:i}$ needs to be formed which would impact the delay. This extra delay can however be eliminated by noting that the critical path for a n-bit adder is in producing the n-1 th bit which can be expressed as:

$S_{n-1} = a_{n-1} \oplus b_{n-1} \oplus G_{n-2:0}$

$= a_{n-1} \oplus b_{n-1} \oplus p_{n-2}H_{n-2:0}$

But $p_{n-2}$ can be computed faster than $H_{n-2:0}$ and so a multiplexer can be used. This is shown in Figure 5b.

$$S_{n-1} = (a_{n-1} \oplus b_{n-1} \oplus p_{n-2})\, H_{n-2:0} + (a_{n-1} \oplus b_{n-1})\, H_{n-2:0}^{c}$$

Although Ling's method is better than the parallel prefix method, it nevertheless has a number of shortcomings. It parallelizes the computation of $G_{j:i}$ as $p_j H_{j:i}$, but one of the functions, $p_j$, is a very simple bit level propagate while the other function, $H_{j:i}$, is much more complex and so the parallelization is very limited. This parallelization, $G_{j:i} = p_j H_{j:i}$ cannot be extended to more than two functions, that is no method is provided to parallelize $G_{j:i}$ as XYZ etc. Ling's method allows for the speed of the first level only (compared to the parallel prefix method) and even this is very limited allowing for at most a reduction in the fan-in of the AND gates at the first level by at most 1. It offers no advantage over parallel prefix method when combining Group functions, in terms of the complexity of the gates and the fan out of Group functions.

The first drawback of Ling's approach is that although the carry function $G_{j:i} = p_j H_{j:i}$ is broken down as a combination of two simpler functions, which can be computed in parallel, one of the functions is a very simple $p_j = a_j + b_j$ while the second is much more complex. Thus the impact on the delay in calculating the carry is very small.

A further prior art technique for generating carry bits is described in US patent number 5,964,827 (IBM Corporation). The IBM technique involves generating $G_{3:0}$ by factorising $p_3 p_2$ out of the expression for $G_{3:0}$. The result is:

$$G_{3:0} = g_3 + p_3 p_2 [g_2 + g_1 + p_1 g_0] = [g_3 + p_3 p_2][g_3 + g_2 + g_1 + p_1 g_0]$$

The function $G_{15:0}$ is then determined using a similar factorisation involving a group function, giving:

$$G_{15:0} = [G_{15:12} + P_{15:12} P_{11:8}][G_{15:12} + G_{11:8} + G_{7:4} + P_{3:0} G_{3:0}].$$

The IBM method provides the advantage that the above factorisation reduces all AND gates to only two inputs. This is particularly useful in dynamic logic implementations because AND gates slow down significantly as the number of inputs is increased. Thus, the aim of the IBM idea is to reduce the number of inputs to a minimum for each AND gate. This can be achieved by combining only four bits at each level to produce a group generate function or a carry, and performing the above factorisation, in which each AND gate has only two inputs. In this type of technology, it is not as crucial to limit the number of inputs on an OR gate. However in the IBM method, the generate function is fully calculated at each stage by performing an AND operation between the two terms in brackets. This is unnecessary, and slows down the circuit.

The present invention uses reduced generate logic which is simpler logic than the generate logic i.e. less logic is required and the computation is faster. The output of generate logic indicates if a carry will be generated out of a group of input bits. The output of reduced generate logic for a group of input bits, partitioned into at least one most significant bit, and at least one least significant bit, is the logical OR of a generate logic for the least significant bits and logic for performing a function X for the most significant bits. X represents a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits.

One aspect of the present invention provides a method and apparatus for forming reduced generate logic for a group of input bits using at least one reduced generate output for at least one subgroup of the group of input bits, at least one reduced generate logic generating an output based on an X function using at least two most significant input bits.

One aspect of the present invention provides a method and apparatus for carry generation in which logic is arranged in levels of logic in which each level computes reduced generate functions, and lower levels compute reduced generate functions from reduced generate functions at higher levels, wherein at least one of the reduced generate functions has an X component ranging over at least two bits. The levels are preferable levels in a tree structure.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising first logic for receiving a plurality of bits of the binary inputs and for generating at least one intermediate output; final logic for receiving at least one intermediate output of the first logic and for generating the carry bit output; wherein said final logic is arranged to generate the carry bit output using a reduced generate function for a group of bits of the binary inputs and at least one intermediate output from said first logic at least one of which is generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein first logic and/or said final logic is arranged to use a reduced generate function in which the group or sub-group of bits of the binary inputs is partitioned so that said at least one most significant bit comprises at least two most significant bits.

Another aspect provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising first logic for receiving a plurality of bits of the binary inputs and for generating at least one

intermediate output; final logic for receiving at least one intermediate output of the first logic and for generating the sum bit output; wherein said final logic is arranged to generate the sum bit output using a reduced generate function for a group of bits of the binary inputs and at least one intermediate output from said first logic at least one of which is generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein first logic and/or said final logic is arranged to use a reduced generate function in which the group or sub- group of bits of the binary inputs is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In this aspect of the present invention, the sum bit is calculated directly using the reduced generate function, rather than generating the carry and logically exclusive OR combining the carry bit with the exclusive OR combination of input bits. In one embodiment the final logic includes at least one multiplexer.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic including a final level of logic for receiving outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of at least one previous level and for generating an intermediate output; and output logic for generating the carry bit

output using at least one of the intermediate outputs from the final level of logic; wherein at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs using intermediate outputs from at least one higher level at least one of which is generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function in which the group or sub-group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In one embodiment further logic is provided for generating an output for a group of most significant bits of the binary inputs which is high if a carry is generated out of the group or if all of the bit level propagate bits for the group are high, wherein said output logic is arranged to generate the carry bit as a function of the logical AND of the output of said further logic and the intermediate output of said final level generated as a reduced generate function for a group of bits.

A second aspect provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic including a final level of logic for receiving outputs of at least one

previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of at least one previous level and for generating an intermediate output; and output logic for generating the sum bit output using at least one of the intermediate outputs from the final level of logic; wherein at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs using intermediate outputs from at least one higher level at least one of which is generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function in which the group or sub-group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In this aspect of the present invention, the sum bit is calculated directly using the reduced generate function, rather than generating the carry and logically exclusive OR combining the carry bit with the exclusive OR combination of input bits. In one embodiment the output logic comprises a multiplexer.

In one embodiment further logic is provided for generating an output for a group of most significant bits of the binary inputs which is high if a carry is generated out of the group or if all of the bit level propagate bits for the group are high, wherein said output logic is arranged to generate the carry bit as a function of the logical AND of

the output of said further logic and the intermediate output of said final level generated as a reduced generate function for a group of bits.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic for receiving outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level and for generating an intermediate output; a final level of logic for receiving at least one of the intermediate outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level of logic and for generating the carry bit output; wherein at least one logic unit of at least one of said further levels of logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs using intermediate outputs from at least one higher level, at least one of said intermediate outputs being generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of at least one of said first or further levels of logic is arranged to generate an intermediate output as a reduced generate function in which the group or sub-group of bits of the binary inputs for

said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

Another aspect provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic for receiving at least one of the intermediate outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level and for generating an intermediate output; a final level of logic for receiving outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level of logic and for generating the sum bit output; wherein at least one logic unit of at least one of said further levels of logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs using intermediate outputs from at least one higher level, at least one of said intermediate outputs being generated as a reduced generate function of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of at least one of said first or further levels of logic is arranged to generate an intermediate output as a reduced generate function in which the group or sub-group of bits of the binary inputs for

said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In this aspect of the present invention, the sum bit is calculated directly using the reduced generate function, rather than generating the carry and logically exclusive OR combining the carry bit with the exclusive OR combination of input bits. In one embodiment the final level of logic includes at least one multiplexer.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising first logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; final logic for receiving at least one intermediate output of the first logic and comprising at least one logic unit for receiving at least one intermediate output from at least one logic unit of the first logic and for generating the carry bit output; wherein at least one logic unit of at least one of said first logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of said first logic is arranged to generate an intermediate output for receipt by said final logic as a reduced generate function in which the group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

Another aspect provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising first logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; final logic for receiving at least one intermediate output of the first logic and comprising at least one logic unit for receiving at least one intermediate output from at least one logic unit of the first logic and for generating the sum bit output; wherein at least one logic unit of at least one of said first logic is arranged to generate an intermediate output as a reduced generate function for a group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; and wherein at least one of said at least one logic unit of said first logic is arranged to generate an intermediate output for receipt by said final logic as a reduced generate function in which the group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In this aspect of the present invention, the sum bit is calculated directly using the reduced generate function, rather than generating the carry and logically exclusive OR combining the carry bit with the exclusive OR combination of input bits. In one embodiment the final logic includes at least one multiplexer.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs

and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level carry generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate a high output if a carry is generated out of the first group of most significant bits of said binary input or if said carry propagate function bits for the most significant bits are all high; second logic for receiving bit level carry generate and propagate function bits for said binary inputs to generate a high output if any of said carry generate function bits for the most significant bits are high or if a carry is generated out of a second group of least significant bits of said binary input; and combining logic for generating the carry bit output by combining outputs of said first and second logic.

Another aspect of the present invention provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level carry generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate a high output if a carry is generated out of the first group of most significant bits of said binary input or if said carry propagate function bits for the most significant bits are all high; second logic for receiving bit level carry generate and propagate function bits for said binary inputs to generate a high output if any of said carry generate function bits for the most significant bits are high or if a carry is generated out of a second group of least significant bits of said binary input; and combining logic for generating the sum bit output by combining outputs of said first and second logic.

In this aspect of the present invention, the sum bit is calculated directly rather than generating the carry and logically exclusive OR combining the carry bit with the

exclusive OR combination of input bits. In one embodiment the combining logic includes at least one multiplexer.

In one embodiment of the present invention, the first logic comprises a plurality of first logic modules, each for receiving bit level carry generate and propagate function bits for subgroups of the first group of at least three most significant bits of the binary inputs to generate a high output if a carry is generated for the subgroup of most significant bits of the binary input or if the carry propagate function bits for the subgroup of most significant bits are all high.

In one embodiment, the second logic comprises a plurality of logic modules for receiving subgroups of the second group of least significant bits of the binary input to generate a carry for each of the subgroups and combining logic for combining the generated carrys.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate an output as a function of a logical OR combination of a carry bit output for the first group of most significant bits of said binary input and a result of a logical AND combination of propagate function bits for the most significant bits; second logic for receiving bit level generate and propagate function bits for said binary inputs to generate an output as a function of a result of a logical OR combination of a carry bit output for a group of least significant bits of said binary inputs and a function B which is high if a carry is generated at any bit position in the most significant bits; and combining logic for generating the carry bit output by combining outputs of said first and second logic.

Another aspect of the present invention provides a logic circuit for generation of a sum bit output by combining two sets of binary inputs, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate an output as a function of a logical OR combination of a carry bit output for the first group of most significant bits of said binary input and a result of a logical AND combination of propagate function bits for the most significant bits; second logic for receiving bit level generate and propagate function bits for said binary inputs to generate an output as a function of a result of a logical OR combination of a carry bit output for a group of least significant bits of said binary inputs and a function B which is high if a carry is generated at any bit position in the most significant bits; and combining logic for generating the sum bit output by combining outputs of said first and second logic.

In this aspect of the present invention, the sum bit is calculated directly rather than generating the carry and logically exclusive OR combining the carry bit with the exclusive OR combination of input bits. In one embodiment the combining logic includes at least one multiplexer.

Another aspect of the present invention provides a binary adder circuit comprising the logic circuit as hereinabove described, and including addition logic comprising exclusive OR logic and multiplexer for determining an addition result including the carry bit for the binary inputs

Another aspect of the present invention provides a comparison logic circuit for comparing two binary inputs comprising the logic circuit as hereinabove described, and including logic for using the carry bit to indicate whether one binary input

represents a binary number less than or more than another binary number represented by the other binary input.

The present invention also encompasses the use of reduced modified generate logic (D) which is simpler logic than modified generate logic. Modified generate logic indicates if a carry is generated out of the addition of inputs plus one. This enables the logic unit D to be broken down and computed in a parallel fashion.

Another aspect provides a logic circuit for generation of a carry bit output by adding two sets of binary inputs plus one, the logic circuit comprising first logic for receiving a plurality of bits of the binary inputs and for generating at least one intermediate output; final logic for receiving at least one intermediate output of the first logic and for generating the carry bit output; wherein said final logic is arranged to generate the carry bit output using a reduced modified generate function for a group of bits of the binary inputs and at least one intermediate output from said first logic at least one of which is generated as a reduced generate function or a reduced modified generate function of a sub-group of bits of the binary inputs; wherein a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; wherein a reduced modified generate function is the logical OR of a modified generate function for the least significant bits and the function X for the most significant bits, where the modified generate function is high if a carry is generated on adding the least significant bits plus one and low if not; wherein said final logic is arranged to use a reduced modified generate function in which the group or sub-group of bits of the binary inputs is partitioned so that said at least one most

significant bit comprises at least two most significant bits and/or said first logic is arranged to generate at least one intermediate output as a reduced generate function or a reduced modified generate function in which the group or sub-group of bits of the binary inputs is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

In one embodiment the reduced modified generate function uses a hyper propagate function (PD) for the group of bits, the hyper propagate function comprises a logical AND combination of the modified generate function (D) for at least one least significant bit of the group of bits and a propagate function (P) for at least one most significant bit of the group of bits, and the propagate function is high if a carry into a group of bits would be propagated out of the group of bits. Thus in this embodiment the function D is parallelised. The hyper propagate function PD can be further parallelised by using at least one hyper propagate function for a sub-group of bits.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs plus one, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic including a final level of logic for receiving outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of at least one previous level and for generating an intermediate output; and output logic for generating the carry bit output using at least one intermediate output from the final level of logic; wherein at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function or a reduced modified generate function for a group of bits of the binary inputs using intermediate outputs from at least one higher level at least one of which is generated as a reduced

generate function or reduced modified generate function group of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; wherein a reduced modified generate function is the logical OR of a modified generate function for the least significant bits and the function X for the most significant bits, where the modified generate function is high if a carry is generated on adding the least significant bits plus one and low if not; and wherein at least one of said at least one logic unit of at least one level of logic is arranged to generate an intermediate output as a reduced generate function or reduced modified generate function in which the group or sub-group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

In one embodiment further logic is provided for generating an output for a group of most significant bits of the binary inputs which is high if a carry is generated out of the group or if all of the bit level propagate bits for the group are high, wherein said output logic is arranged to generate the carry bit as a function of the logical AND of the output of said further logic and the intermediate output of said final level generated as a reduced modified generate function for a group of bits.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs plus one, the logic circuit comprising a first level of logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; at least one further level of logic for receiving at least one intermediate output of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level and for generating an intermediate output; a final level of logic for receiving outputs of at least one previous level of logic and comprising at least one logic unit for receiving the intermediate outputs from at least one logic unit of the at least one previous level of logic and for generating the carry bit output; wherein at least one logic unit of at least one of said further levels of logic is arranged to generate an intermediate output as a reduced generate function or a reduced modified generate function for a group of bits of the binary inputs using at least one intermediate output from at least one higher level, at least one of said intermediate outputs being generated as a reduced generate function or a reduced modified generate function of a sub-group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group or sub-group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; wherein a reduced modified generate function is the logical OR of a modified generate function for the least significant bits and the function X for the most significant bits, where the modified generate function is high if a carry is generated on adding the least significant bits plus one and low if not; and wherein at least one of said at least one logic unit of at least one of said first or further levels of logic is arranged to generate an intermediate output as a reduced generate function or

reduced modified generate function in which the group or sub-group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

Another aspect provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs plus one, the logic circuit comprising first logic comprising a plurality of logic units, each logic unit for receiving a plurality of bits of the binary inputs and for generating an intermediate output; final logic for receiving at least one intermediate output of the first logic and comprising at least one logic unit for receiving at least one intermediate output from at least one logic unit of the first logic and for generating the carry bit output; wherein at least one logic unit of at least one of said first logic is arranged to generate an intermediate output as a reduced generate function or a reduced modified generate function for a group of bits of the binary inputs; wherein an intermediate output generated as a reduced generate function for a group of bits, partitioned into at least one most significant bit and at least one least significant bit, is the logical OR of a generate function for the least significant bits and a function X for the most significant bits, where the generate function is high if a carry is generated out of the least significant bits and low if not, and X is a function which is high if a carry is generated out of the most significant bits, low if no carry is generated at any bit position in the most significant bits, and in a don't care state if a carry is generated at some bit position in the most significant bits but no carry is generated out of the most significant bits; wherein a reduced modified generate function is the logical OR of a modified generate function for the least significant bits and the function X for the most significant bits, where the modified generate function is high if a carry is generated on adding the least significant bits plus one and low if not; and wherein at least one of said at least one logic unit of said first logic is arranged to generate an intermediate output for receipt by said final logic as a reduced generate function or a

reduced modified generate function in which the group of bits of the binary inputs for said at least one logic unit is partitioned so that said at least one most significant bit comprises at least two most significant bits.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs plus one, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level carry generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate a high output if a carry is generated out of the first group of most significant bits of said binary input or if said carry propagate function bits for the most significant bits are all high; second logic for receiving bit level carry generate and propagate function bits for said binary inputs to generate a high output if any of said carry generate function bits for the most significant bits are high or if a carry is generated out of a second group of least significant bits plus one of said binary input; and combining logic for generating the carry bit output by combining outputs of said first and second logic.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

In one embodiment of the present invention, the first logic comprises a plurality of first logic modules, each for receiving bit level carry generate and propagate function bits for subgroups of the first group of at least three most significant bits of the binary inputs to generate a high output if a carry is generated for the subgroup of

most significant bits of the binary input or if the carry propagate function bits for the subgroup of most significant bits are all high.

In one embodiment, the second logic comprises a plurality of logic modules for receiving subgroups of the second group of least significant bits of the binary input to generate a carry for each of the subgroups and combining logic for combining the generated carrys.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs plus one, the logic circuit comprising: bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate an output as a function of a logical OR combination of a carry bit output for the first group of most significant bits of said binary input and a result of a logical AND combination of propagate function bits for the most significant bits; second logic for receiving bit level generate and propagate function bits for said binary inputs to generate an output as a function of a result of a logical OR combination of a carry bit output for a group of least significant bits plus one of said binary inputs and a function B which is high if a carry is generated at any bit position in the most significant bits; and combining logic for generating the carry bit output by combining outputs of said first and second logic.

In another aspect of the present invention the sum bit for two inputs plus one can similarly be computed.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output by combining two sets of binary inputs, the logic circuit comprising logic for receiving a plurality of bits of the binary inputs and for generating the carry

bit output; wherein said logic is arranged to generate the carry bit output as the logical AND of a generate function G for at least one most significant bit, a reduced modified generate function for the said at least one most significant bit and at least one middle bit of the binary inputs and a reduced generate function for said at least one middle bit and at least one least significant bit of the binary inputs; wherein said reduced generate function is the logical OR of a generate function G for the at least one least significant bit and a function X for the at least one most significant bit and the at least one middle bit, where the generate function for the at least one least significant bit is high if a carry is generated out of the at least one least significant bit and low if not, and X is a function which is high if a carry is generated out of the at least one most significant bit and said at least one middle bit, low if no carry is generated at any bit position in the at least one most significant bit and said at least one middle bit, and in a don't care state if a carry is generated at some bit position in the at least one most significant bit and said at least one middle bit but no carry is generated out of the at least one most significant bit and said at least one middle bit; said reduced modified generate function is the logical OR of a modified generate function D for the at least one middle bit and the function X for the most significant bits, where the modified generate function D for the at least one middle bit is high if a carry is generated on adding the at least one middle bit plus one and low if not.

Another aspect of the present invention provides a logic circuit for generation of a carry bit output D by combining two sets of binary inputs plus 1, the logic circuit comprising logic for receiving a plurality of bits of the binary inputs and for generating the carry bit output; wherein said logic is arranged to generate the carry bit output as the logical AND of a modified generate function D for at least one most significant bit, a first reduced modified generate function for the said at least one most significant bit and at least one middle bit of the binary inputs and a second reduced modified generate function for said at least one middle bit and at least one least significant bit of the binary inputs; wherein said second reduced modified generate function is the logical OR of a modified generate function for the at least one least significant bit and a function X for the at least one most significant bit and

the at least one middle bit, where the modified generate function for the at least one least significant bit is high if a carry is generated out of the at least one least significant bit plus one and low if not, and X is a function which is high if a carry is generated out of the at least one most significant bit and said at least one middle bit, low if no carry is generated at any bit position in the at least one most significant bit and said at least one middle bit, and in a don't care state if a carry is generated at some bit position in the at least one most significant bit and said at least one middle bit but no carry is generated out of the at least one most significant bit and said at least one middle bit; said first reduced modified generate function is the logical OR of a modified generate function for the at least one middle bit and the function X for the most significant bits, where the modified generate function for the at least one middle bit is high if a carry is generated on adding the at least one middle bit plus one and low if not.

Another aspect of the present invention provides a method of designing a logic circuit for generating a carry bit or sum bit from the combination of two j-bit binary inputs, the method comprising: performing a first parallelisation of the function $G_{j-1:0}$ for generating the carry in accordance with a first relationship $G_{a:c} = D_{a:b}(X_{a:b} + G_{b-1:c})$ to generate a parallelised function $D_{j-1:k}(X_{j-1:k} + G_{k-1:0})$, where G represents a generate function for a group of bits from j-1 to 0 or from k-1 to 0, D represents a logical OR of a generate function and a propagate function for a group of bits from j-1 to k, and X represents a function which is high if a carry is generated out of the j-1 to k bits, low if no carry is generated at any bit position in the j-1 to k bits, and in a don't care state if a carry is generated at some bit position in the j-1 to k bits but no carry is generated out of the j-1 to k bits; performing a second parallelisation of the generate function of the parallelised function using a parallel prefix method to generate a further parallelised function; and designing a logic circuit in accordance with the further parallelised function.

In one embodiment the method includes performing a further parallelisation of the further parallelised function using the first relationship to parallelise the generate function for a group of least significant bits.

In one embodiment the method includes performing a further parallelisation of the further parallelised function using a parallel prefix method to parallelise the further parallelised generate function for a group of least significant bits.

In one embodiment the method includes repeatedly performing further parallelisations of the further parallelised function using alternately the first relationship and a parallel prefix method to parallelise the generate function for a group of least significant bits.

In one embodiment the method includes performing a parallelisation of D using a third relationship $D_{a:c} = D_{a:b} (X_{a:b} + D_{b-1:c})$ to generate a further parallelised function for use in the logic design.

In one embodiment the method includes performing a further parallelisation of D in the further parallelised function using a parallel prefix method.

In one embodiment the method includes repeatedly performing further parallelisations of D in the further parallelised function using alternately the third relationship and a parallel prefix method to parallelise D.

In one embodiment of the present invention the method includes using at least one multiplexer in conjunction with logic for performing the further parallelised functions.

The present invention allows for a greater degree of parallelisation than in either Ling or IBM, thus speeding up the computation of carry and/or sum bits.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 shows a prior art logic circuit for generating a carry bit using single bit generate and single bit propagate functions;

Figure 2 shows a prior art logic circuit for generating a carry bit using the Parallel Prefix Method;

Figure 3 shows a prior art logic circuit for generating the most significant carry bit in a 9 bit addition, using the base 3 Parallel Prefix method;

Figure 4 shows a prior art logic circuit for generating a carry bit using the Ling method;

Figure 5a shows a prior art logic circuit for generating the most significant carry bit in a 9 bit addition, using the Ling method combined with the base 3 Parallel Prefix method;

Figure 5b shows a prior art logic circuit in which the Ling method is used to move an XOR gate off the critical path;

Figure 6 shows a representation of the data structure of a j+1 bit addition, and the derivation of intermediate functions $X_{j:k}$, $D_{j:k}$, $G_{k-1:0}$ and output $G_{j:0}$;

Figure 7 shows a logic circuit according to an embodiment of the invention in which the functions $X_{j:k}$, $D_{j:k}$, $G_{k-1:0}$ are implemented using logic gates and combined to produce an output of $G_{j:0}$;

Figure 8a shows a representation of the data structure of a j+1 bit addition, and the derivation of intermediate functions $X_{j:k}$, $D_{j:k}$, $D_{k-1:0}$ and $D_{j:0}$;

Figure 8b shows a logic circuit in which the factorisation $D_{n-2:k}[X_{n-2:k} + G_{k-1:0}]$ is used to allow an XOR gate to be moved off the critical path;

Figure 8c shows a logic circuit in which the factorisation $D_{n-2:k'}[X_{n-2:k'} + D_{k'-1:k}][X_{n-2:k} + D_{k-1:0}]$ is used to allow an XOR gate to be moved off the critical path;

Figure 8d shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $X_{n-1:k} + G_{k-1:k'}$, $X_{n-1:k} + G_{k-1:k'}$, and $P_{k-1:k'}D_{k'-1:k''}$ and $D_{n-1:k}$;

Figure 8e shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $X_{n-1:k}$, $X_{k-1:m} + G_{m-1:k'}$, $X_{k'-1:m'} + G_{m'-1:0}$, and $P_{m-1:k'}D_{k'-1:m'}$ and $D_{n-1:m}$;

Figure 8f shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $X_{n-1:k}$, $X_{k-1:k'}$, $X_{k'-1:m} + G_{m-1:0}$ and $D_{n-1:m}$;

Figure 9 shows a logic circuit according to an embodiment of the invention in which the functions $D_{8:5}$, $B_{8:5}$, $G_{4:0}$ are implemented using logic gates and combined to produce an output of $G_{8:0}$;

Figure 10 shows a logic circuit according to an embodiment of the invention, in which the functions $B_{8:5}$, $G_{4:3}$, $P_{4:3}$ and $G_{2:0}$ are implemented using logic gates and combined to produce an output of $G_{8:0}$;

Figure 11 shows a logic circuit according to an embodiment of the invention, in which the functions $B_{8:6}$, $B_{5:5} + G_{4:3}$, $P_{4:3}D_{2:2}$ and $B_{2:2} + G_{2:0}$ are implemented using logic gates and combined to produce an output of $G_{8:0}$;

Figure 12 shows a ternary tree implementation of a final carry generator on a 9-bit adder, in which the term $D_{8:5}$ is generated using already pre-formed building blocks;

Figure 13 shows a representation of the data structure of a n bit addition, and the derivation of intermediate functions $P_{n-1:k}$, $P_{k-1:m}D_{m-1:k'}$, $P_{k'-1:m'}D_{m'-1:0}$, $B_{m-1:k'} + G_{k'-1:m'}$ and $D_{n-1:m}$;

Figure 14a shows a representation of the structure of functions calculated at different levels of an adder according to an embodiment of the invention;

Figure 14b shows a representation of the structure of functions calculated at different levels of an adder according to an embodiment of the invention;

Figure 14c shows a representation of the structure of functions calculated at different levels of an adder according to an embodiment of the invention;

Figure 14d shows a representation of the structure of functions calculated at different levels of an adder according to an embodiment of the invention;

Figure 15 shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $X_{n-1:k}$, $X_{k-1:m} + G_{m-1:k'}$, $X_{k'-1:m'} + G_{m'-1:k''}$, $X_{k''-1:m''} + G_{m''-1:0}$, $P_{m-1:k'}D_{k'-1:m'}$, $P_{m'-1:k''}D_{k''-1:m'}$ and $D_{n-1:m}$;

Figure 16 shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $P_{n-1:k}$, $P_{k-1:m}D_{m-1:k'}$, $P_{k'-1:m'}D_{m'-1:k''}$, $P_{k''-1:m''}D_{m''-1:0}$, $B_{m-1:k'} + G_{k'-1:m'}$ and $B_{m'-1:k''} + G_{k''-1:m'}$;

Figure 17 shows a representation the data structure of a n bit addition, and the derivation of intermediate functions $X_{n-1:k}$, $X_{k-1:k'}$, $X_{k'-1:m} + G_{m-1:k''}$, $X_{k''-1:m'} + G_{m'-1:0}$, $P_{m-1:k''}D_{k''-1:m'}$, and $D_{n-1:m}$; and

Figure 18 shows a logic circuit according to an embodiment of the invention, for a 16 bit adder.

As a first embodiment to the invention there is disclosed a method which allows for the carry to be formed as a combination of functions, which can be computed in parallel, each of which is more complex than a simple single bit-level propagate.

An embodiment of the invention will now be illustrated by way of example. Consider

$$G_{4:0} = g_4 + p_4g_3 + p_4p_3g_2 + p_4p_3p_2g_1 + p_4p_3p_2p_1g_0$$

Ling's approach breaks this as

$$G_{4:0} = p_4[g_4 + g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0]$$

The inventors have observed that the delay of the carry term can be reduced significantly by increasing the delay of some other term by more than a simple bit-level propagate. For example:

$$G_{4:0} = [g_4 + p_4p_3][g_4 + g_3 + g_2 + p_2g_1 + p_2p_1g_0]$$

$$G_{4:0} = [g_4 + p_4g_3 + p_4p_3p_2][g_4 + g_3 + g_2 + g_1 + p_1g_0]$$

The inventors have further observed that the delay of the carry term can be reduced significantly by increasing the delay of some other terms, rather than just one. For example:

$$G_{4:0} = p_4[g_4 + p_3][g_4 + g_3 + g_2 + p_2g_1 + p_2p_1g_0]$$

$$G_{4:0} = p_4[g_4 + g_3 + p_3p_2][g_4 + g_3 + g_2 + g_1 + p_1g_0]$$

$G_{4:0} = [g_4 + p_4p_3][g_4 + g_3 + p_2][g_4 + g_3 + g_2 + g_1 + p_1g_0]$

In the following a Logic unit indicating carry generation from addition of 2 numbers $a_j \ldots a_k$ and $b_j \ldots b_k$ plus 1 will be denoted by:

$D_{j:k} = G_{j:k} + P_{j:k}$

The inventors have observed that in general

$G_{j:0} = D_{j:k}[X_{j:k} + G_{k-1:0}]$

Thus a method and apparatus are disclosed, as shown in Figure 6, for a Logic unit indicating carry generation of addition, the input bits being divided into two groups, least significant and most significant bits, in which:

$G_{k-1:0}$ denotes a logic unit indicating carry generation from addition of least significant bits.

$D_{j:k}$ denotes a logic unit indicating carry generation from addition of most significant bits plus 1.

$X_{j:k}$ denotes a logic unit which is high when a carry is generated out of the most significant bits, and is low if no carry is generated at any bit position in the most significant bits. The unit is in a don't care state if a carry is generated at some bit position but no carry is generated out of the most significant bits.

The outputs of the above three logic units are combined in a logical unit for generating $G_{j:0}$.

For 2-bit addition, the following Karnaugh maps respectively illustrate the logic unit X, logic when a carry is generated, and logic when no carry is generated at any bit position.

| $a_1b_1/a_0b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | Don't care | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 0 |

| $a_1b_1/a_0b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | **1** | 0 |
| 11 | **1** | **1** | **1** | **1** |
| 10 | 0 | 0 | **1** | 0 |

| $a_1b_1/a_0b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | **0** | **0** | 1 | **0** |
| 01 | **0** | **0** | 1 | **0** |
| 11 | 1 | 1 | 1 | 1 |
| 10 | **0** | **0** | 1 | **0** |

The inventors have observed that the simplest implementation of the $X_{j:i}$ unit is

$B_{j:i} = g_j + g_{j-1} + \ldots + g_{i+1} + g_i$

Thus, as illustrated in Figure 7:

$G_{j:0} = D_{j:k}[B_{j:k} + G_{k-1:0}]$

Example:

$G_{8:0} = D_{8:5}[B_{8:5} + G_{4:0}]$

The inventors have observed that logic indicating carry generation of the addition of two numbers plus 1 can be parallelized as:

$$D_{j:0} = D_{j:k}[X_{j:k} + D_{k-1:0}]$$

Thus a method and logic circuit are disclosed, and illustrated in figure 8a, for carry generation in for example addition, in which the input bits are divided into 2 groups, least significant and most significant bits. The logic circuit comprises:

A logic unit indicating carry generation from addition of least significant bits plus 1, denoted by $D_{k-1:0}$.

A logic unit indicating carry generation from addition of most significant bits plus 1, denoted by $D_{j:k}$.

A logic unit which is high when a carry is generated out of the most significant bits, and is low if no carry is generated at any bit position in the most significant bits. The unit is in a don't care state if a carry is generated at some bit position but no carry is generated out of the most significant bits, denoted by $X_{j:k}$.

A logical unit for combining outputs of the 3 logic units.

The inventors have observed that in such a parallelization the simplest implementation of the $X_{j:i}$ unit is

$$B_{j:i} = g_j + g_{j-1} + \ldots + g_{i+1} + g_i$$

Thus $D_{j:0} = D_{j:k}[B_{j:k} + D_{k-1:0}]$

The inventors have further observed that further parallelization of carry generation can be achieved by repeated use of parallelizing D

$$G_{j:0} = D_{j:k'}[X_{j:k'} + D_{k'-1:k}][X_{j:k} + G_{k-1:0}]$$

Example: $G_{15:0} = D_{15:8}[B_{15:8} + D_{7:4}][B_{15:4} + G_{3:0}]$

An adder does have the problem that to produce the actual carry out the logical AND of D and X + G needs to be formed which would impact the delay. This extra delay can however be eliminated by noting that the critical path for the n th bit of an adder is:

$$S_{n-1} = a_{n-1} \oplus b_{n-1} \oplus G_{n-2:0}$$
$$= a_{n-1} \oplus b_{n-1} \oplus D_{n-2:k}[X_{n-2:k} + G_{k-1:0}]$$

By choosing an appropriate k, $D_{n-2:k}$ can be computed faster than $X_{n-2:k} + G_{k-1:0}$ and so a multiplexer can be used. This is illustrates in figure 8b.

$$S_{n-1} = (a_{n-1} \oplus b_{n-1} \oplus D_{n-2:k})\,[X_{n-2:k} + G_{k-1:0}] + (a_{n-1} \oplus b_{n-1})\,[X_{n-2:k} + G_{k-1:0}]^c$$

This method can be applied to the invention when G is produced as a combination of more than two functions, by using more than one multiplexer as illustrated in figure 8c.

$$S_{n-1} = ((a_{n-1} \oplus b_{n-1} \oplus D_{n-2:k'})\,[X_{n-2:k'} + D_{k'-1:k}] + (a_{n-1} \oplus b_{n-1})\,[X_{n-2:k'} + D_{k'-1:k}]^c)\,[X_{n-2:k} + G_{k-1:0}] + (a_{n-1} \oplus b_{n-1})\,[X_{n-2:k} + G_{k-1:0}]^c$$

In another embodiment of the invention the inventors have realised that the parallelization of carry generation as disclosed above can be combined with the parallelization provided by the parallel prefix method to determine carries or building blocks in a tree like structure and provide further speed up of carry generation. The inventors have further realized that there are several methods for this combination, the best combination depending on type of technology being used, for

example static CMOS and dynamic circuits among others. The best combination will become apparent to those skilled in the art.

This method of combining will now be illustrated.

The parallel prefix method provides the following parallelizations:

$$G_{j:i} = G_{j:k} + P_{j:k}G_{k-1:i}$$

$$D_{j:i} = G_{j:k} + P_{j:k}D_{k-1:i}$$

To implement $G_{n-1:0}$ we can first parallelize using the method disclosed above:

$$G_{n-1:0} = D_{n-1:k}[X_{n-1:k} + G_{k-1:0}]$$

The inventors have observed that since $X_{n-1:k} + G_{k-1:0}$ is an OR of two terms and the parallel prefix method provides a means of parallelizing $G_{k-1:0}$ also as an OR of two terms, it is well known that an OR-OR combination can be reduced to a single OR combination and in many technologies gates with 3 or 4 inputs can be implemented efficiently, the combination of the two methods results in:

$$X_{n-1:k} + G_{k-1:0} = X_{n-1:k} + G_{k-1:k'} + P_{k-1:k'}G_{k'-1:0}$$

The inventors have realized that further parallelization can be achieved by parallelizing $G_{k'-1:0}$ by the method of the current invention to get an AND-AND combination which can be reduced to a single AND combination and the efficiency of larger input gates can be used.

Thus parallelizing $G_{k'-1:0}$ as

$$G_{k'-1:0} = D_{k'-1:k''}[X_{k'-1:k''} + G_{k''-1:0}]$$

We arrive at:

$$X_{n-1:k} + G_{k-1:0} = [X_{n-1:k} + G_{k-1:k'}] + [P_{k-1:k'}\, D_{k'-1:k''}][X_{k'-1:k''} + G_{k''-1:0}]$$

This is illustrate in figure 8d.

The benefits of this method will now be illustrated by way of example:

$$G_{7:0} = D_{7:6}[B_{7:6} + G_{5:0}]$$

$$B_{7:6} + G_{5:0} = [B_{7:6} + G_{5:4}] + P_{5:4}G_{3:0}$$
$$= [B_{7:6} + G_{5:4}] + [P_{5:4}D_{3:2}][B_{3:2} + G_{1:0}]$$

The building blocks are given by:

$$D_{7:6} = g_7 + p_7p_6$$

$$B_{7:6} + G_{5:4} = g_7 + g_6 + g_5 + p_5g_4$$

$$B_{3:2} + G_{1:0} = g_3 + g_2 + g_1 + p_1g_0$$

$$P_{5:4}D_{3:2} = p_5p_4[g_3 + p_3p_2] = p_5p_4p_3[g_3 + p_2]$$

We now compare this to Ling's method:

$$G_{7:0} = p_7 H_{7:0}$$

$$H_{7:0} = H_{7:4} + P_{6:3} H_{3:0}$$

In which the building blocks are given by:

$$P_{6:3} = p_6p_5p_4p_3$$

$$H_{7:4} = g_7 + g_6 + p_6g_5 + p_6p_5g_4$$

$$H_{3:0} = g_3 + g_2 + p_2g_1 + p_2p_1g_0$$

Notice that $B_{3:2} + G_{1:0}$ is simpler than $H_{3:0}$, $B_{7:6} + G_{5:4}$ is simpler than $H_{7:4}$ but $P_{5:4}D_{3:2}$ and $D_{7:6}$ are more complex than $P_{6:3}$ and $p_7$ respectively. But the critical path of the current method is shorter. Moreover, the implementation according to this embodiment of the invention has less fan-out, $p_6$ has a fan-out of three in Ling's method but the maximum fan-out of a signal in the current embodiment is two.

The method disclosed above applies to binary trees. It will now be shown that further speed up of carry generation can be achieved by combining more that two terms. The method and apparatus will be illustrated by way of ternary trees.

We first illustrate Ling's method on ternary trees and point out the shortcomings.

The starting point of this method is to parallelize carry generation as:

$$G_{n-1:0} = p_{n-1} H_{n-1:0}$$

Then

$$H_{n-1:0} = g_{n-1} + G_{n-2:0} = g_{n-1} + G_{n-2:k'} + P_{n-2:k'} G_{k'-1:k''} + P_{n-2:k'} P_{k'-1:k''} G_{k''-1:0}$$

By applying $G_{j:i} = p_j H_{j:i}$ to $G_{k'-1:k''}$ and $G_{k''-1:0}$ we have

$$H_{n-1:0} = H_{n-1:k'} + P_{n-2:k'-1} H_{k'-1:k''} + P_{n-2:k'} P_{k'-1:k''-1} H_{k''-1:0}$$

This has the form A + BC + DEF

But notice that although H is a little simpler than G this method offers no advantage over the parallel prefix method when three of the H's are combined. Also note that although ternary trees have fewer levels than binary trees, in this method each level is much more complex then the binary tree parallel prefix method. Moreover very

high fan-out results. Thus the ternary tree method offers little if any advantage over the prior art binary method.

Method and apparatus are now disclosed which overcome these shortcomings. We divide the n input bits into three segments, as shown in figure 8e: [n-1,k], [k-1,k'] and [k'-1,0].

By choosing an m lying in the middle segment we have:

$$G_{n-1:0} = D_{n-1:m}[X_{n-1:m} + G_{m-1:0}]$$

We now consider

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:m} + G_{m-1:k'} + P_{m-1:k'}\, G_{k'-1:0}$$

By choosing an m' lying in the third segment and parallelizing $G_{k'-1:0}$ as $D_{k'-1:m'}[X_{k'-1:m'} + G_{m'-1:0}]$ according to the method of one embodiment of the current invention we have computed $X_{n-1:m} + G_{m-1:0}$ in terms of three smaller terms of the same form.

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:k} + [X_{k-1:m} + G_{m-1:k'}] + [P_{m-1:k'}\, D_{k'-1:m'}]\, [X_{k'-1:m'} + G_{m'-1:0}]$$

Note that this logic combination is a simple $K_2 + K_1 + Q_0K_0$ compared to $H_2 + P_2H_1 + P_2P_1H_0$ for Ling's method. This has been achieved at the expense of a more complex D since it ranges from n-1 to m. Those skilled in the art can choose an appropriate m such that the critical path for the two units $D_{n-1:m}$ and $[X_{n-1:m}+ G_{m-1:0}]$ is balanced in a manner that results in faster carry generation depending on the technology.

The inventors have observed that this method is very advantageous in Field Programmable Gate Array technology. It is known that in this technology LUTs (Look-Up Tables) (i.e. look-up table based FPGAs) are provided which can compute

any logic function, a very common choice for the number of variables which can be input to an LUT is four variables. It is noted that $K_2 + K_1 + Q_0K_0$ is a function of four variables where as $H_2 + P_2H_1 + P_2P_1H_0$ is a function of five variables.

Notice that if m had been chosen to lie in the third segment, then $D_{n-1:m}$ would have become more complex still but resulting in simpler:

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:k} + X_{k-1:k'} + [X_{k'-1:m} + G_{m-1:0}]$$

This is illustrated in figure 8f.

Note that the parallelization of $D_{n-1:m}$ can be carried out in a similar manner. This is illustrated in figure 8f.

Figures 9, 10 & 11 show a sequence of steps to derive the ternary tree implementation of the final carry generation in a 9-bit adder.

$$G_{8:0} = D_{8:5}[B_{8:5} + G_{4:0}] \quad \text{(Figure 9)}$$

$$= D_{8:5}[B_{8:5} + G_{4:3} + P_{4:3}G_{2:0}] \quad \text{(Figure 10)}$$

$$= D_{8:5}[B_{8:6} + [B_{5:5} + G_{4:3}] + P_{4:3}D_{2:2}[B_{2:2} + G_{1:0}]] \quad \text{(Figure 11)}$$

The inventors have observed that logic can be shared in the implementations of $D_{n-1:m}$ and $[X_{n-1:m} + G_{m-1:0}]$.

This is now illustrated by way of example.

$$D_{8:5} = G_{8:6} + P_{8:5}$$

$$= p_8[B_{8:8} + G_{7:6} + P_{7:5}]$$

and $B_{8:8} + G_{7:6}$ is a suitable $X_{8:6}$ which can replace $B_{8:6}$ in the above parallelization of $G_{8:0}$. This sharing of logic allows for the reduction of silicon area. The complete parallelization of $G_{8:0}$ according to the invention is shown in figure 12.

We have thus far disclosed how a term of the form $X_{n-1:m} + G_{m-1:0}$ can be constructed out of terms over a smaller range, for example in the ternary tree method.

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:k} + [X_{k-1:m} + G_{m-1:k'}] + [P_{m-1:k'} D_{k'-1:m'}] [X_{k'-1:m'} + G_{m'-1:0}]$$

Note that each of the terms $X_{n-1:k}$, $X_{k-1:m} + G_{m-1:k'}$, $X_{k'-1:m'} + G_{m'-1:0}$ can be constructed in the same manner from terms over even smaller ranges. We have disclosed a recursive method for forming X + G over a range in terms of X + G over a smaller range in a tree structure. However this involves the PD term. A method is now disclosed for forming the PD term over a range in terms of X + G and PD terms over a smaller range.

Before illustrating this method we fix some notation:

By underlining a logic unit we mean the non-underlined logic unit but with complemented inputs.

The inventors have observed the following relationships between $G_{j:i}$ and $D_{j:i}$.

$G_{j:i} = \underline{D_{j:i}}^c$

$G_{j:i}^c = \underline{D_{j:i}}$

$\underline{G_{j:i}}^c = D_{j:i}$

$\underline{G_{j:i}} = D_{j:i}^c$

Also it is easy to see that

$P_{j:i} = \underline{B_{j:i}}^c$

$P_{j:i}^c = \underline{B_{j:i}}$

$\underline{P_{j:i}}^c = B_{j:i}$

$\underline{P_{j:i}} = B_{j:i}^c$

Thus

$$P_{n-1:m}D_{m-1:0} = \underline{[B_{n-1:m} + G_{m-1:0}]^c}$$

$$= \underline{(B_{n-1:k} + [B_{k-1:m} + G_{m-1:k'}] + [P_{m-1:k'} D_{k'-1:m'}] [B_{k'-1:m'} + G_{m'-1:0}])^c}$$

$$= \underline{B_{n-1:k}^c [B_{k-1:m} + G_{m-1:k'}]^c ([P_{m-1:k'} D_{k'-1:m'}]^c + [B_{k'-1:m'} + G_{m'-1:0}]^c)}$$

$$= P_{n-1:k}[P_{k-1:m}D_{m-1:k'}] ([B_{m-1:k'} + G_{k'-1:m'}] + [P_{k'-1:m'}D_{m'-1:0}])$$

Note that this logic combination is a simple $Q_2Q_1(K_0 + Q_0)$. The process is illustrated in figure 13. We have now disclosed a method and apparatus for recursively constructing X + G and PD in a tree structure. Figure 14a shows the tree structure for the carry out of a 27-bit adder. Figure 14b shows the tree structure for the carry out of a 27-bit adder in a form from which those knowledgeable in the art can derive a silicon layout of an adder. Figure 14c shows the tree structure for the carry out of a 32-bit adder according to the present invention. Figure 14d shows the tree structure for the carry out of a 32-bit adder to aid the layout process.

The inventors have observed that this method is very advantageous in Field Programmable Gate Array technology. It is known that in this technology LUTs are provided which can compute any logic function, a very common choice for the number of variables which can be input to an LUT is four variables. It is noted that $Q_2Q_1(K_0 + Q_0)$ is a function of four variables.

The inventors have observed that this method can be applied to higher order trees such as quaternary, quintic and so forth. The quaternary method will now be illustrated.

We divide the n input bits into three segments:

[n-1,k], [k-1,k'], [k'-1,k''] and [k''-1,0]. By choosing a suitable m in the second segment we can parallelize $G_{n-1:0}$ as

$$G_{n-1:0} = D_{n-1:m}[X_{n-1:m} + G_{m-1:0}]$$

We now construct $X_{n-1:m} + G_{m-1:0}$ out of four smaller segments. Appropriate m' and m'' are chosen in the third and fourth segments respectively. The $G_{m-1:0}$ is parallelized according to the parallel prefix method.

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:m} + G_{m-1:k'} + P_{m-1:k'} G_{k'-1:k''} + P_{m-1:k'} P_{k'-1:k''} G_{k''-1:0}$$

The terms $G_{k'-1:k''}$ and $G_{k''-1:0}$ are now parallelized according to the method of the current invention.

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:k} + [X_{k-1:m} + G_{m-1:k'}] + P_{m-1:k'} D_{k'-1:m'} [X_{k'-1:m'} + G_{m'-1:k''}] + P_{m-1:k'} P_{k'-1:k''} D_{k''-1:m''} [X_{k''-1:m''} + G_{m''-1:0}]$$

The inventors have further observed that $P_{m-1:k'}$ can be replaced by $P_{m-1:k'} D_{k'-1:m'}$ thus allowing for sharing of logic and so reducing area. This is illustrated in figure 15.

The method of constructing PD in a quaternary tree can be derived as before

$$P_{n-1:m} D_{m-1:0} = P_{n-1:k}[P_{k-1:m} D_{m-1:k'}][[B_{m-1:k'} + G_{k'-1:m'}] + [P_{k'-1:m'} D_{m'-1:k''}]] [[B_{m-1:k'} + [B_{k'-1:k''} + G_{k''-1:m''}][P_{k''-1:m''} D_{m''-1:0}]]$$

This is illustrated in figure 16 and has the form $Q_3Q_2[K_2 + Q_1][K_2 + K_1 + Q_0]$

As with the ternary method, m can be chosen in different segments. The further to the least significant segment results in a less complex K = X + G but a more

complex D. This aspect of the invention will now be illustrated. If for the quaternary method we choose m to lie in the third segment then:

$$X_{n-1:m} + G_{m-1:0} = X_{n-1:k} + X_{k-1:k'} + [X_{k'-1:m} + G_{m-1:k''}] + [P_{m-1:k''}D_{k''-1:m'}][X_{k''-1:m'} + G_{m'-1:0}]$$

This is illustrated in figure 17 and has the form $K_3 + K_2 + K_1 + Q_1K_0$. Notice that a PD term can also be constructed having the form $Q_3Q_2Q_1[K_1 + Q_0]$

Figure 18 shows a quaternary tree implementation of the final carry generation in a 16-bit adder. This example in particular illustrates that the final Generate function is AND of at least three terms and the first level is not Ling.

$$G_{15:0} = D_{15:6}K_{15:0} = D_{15:10}J_{15:6}K_{15:0}$$

The building blocks of the construction will now be considered.

It has been decided in this example that the sixteen bits be divided into 4 groups and it is decided that the maximum complexity of the functions at the first level should be $K_2+K_1+K_0+Q_1K_0$.

$$K_{3:0} = g_3 + g_2 + g_1 + p_1g_0$$
$$K_{7:4} = g_7 + g_6 + g_5 + p_5g_4$$
$$K_{11:8} = g_{11} + g_{10} + g_9 + p_9g_8$$
$$K_{15:12} = g_{15} + g_{14} + g_{13} + p_{13}g_{12}$$

$$K_{15:0} = K_{15:12} + K_{11:8} + K_{7:4} + [P_{5:4}D_{3:2}]K_{3:0}$$

Note $K_{15:0}$ has the form $K_2 + K_1 + K_0 + Q_1K_0$

$$P_{5:4}D_{3:2} = p_5p_4\,p_3[g_3 + p_2]$$

Which has the form $Q_2Q_1Q_0[K_1 + Q_0]$

We now construct the D term:

$D_{15:6} = D_{15:10}J_{15:6}$

$J_{15:6} = K_{15:12} + K_{11:8} + P_{9:8}D_{7:6}$

$P_{9:8}D_{7:6} = p_9p_8p_7[g_7 + p_6]$

We need to construct the new term:

$D_{15:10} = D_{15:14}[K_{15:12} + P_{13:12}D_{11:10}]$

$P_{13:12}D_{11:10} = p_{13}p_{12}\,p_{11}[g_{11} + p_{10}]$

$D_{15:14} = p_{15}\,[g_{15} + p_{14}\,]$

The inventors have observed the parallelizations $G = D_2[X_2 + G_1]$ and $G = G_2 + P_2G_1$ can be used in many different combinations to derive optimal implementations depending on the type of technology e.g. Static CMOS, dynamic circuits etc.

The following 16-bit example shows a different logical combination, which is suitable for dynamic circuit techniques. It is known that in dynamic circuit implementations wide OR gates can be implemented efficiently but wide AND gates are slow in comparison. The inventors have further observed that the critical path of a 16-bit adder is in forming the $a_{14} \oplus b_{14} \oplus G_{14:0}$. The inventors have further observed that implementation of D results in a faster circuit than G. The inventors have further observed that if inverted primary inputs are available then

$a_{14} \oplus b_{14} \oplus G_{14:0} = a_{14} \oplus b_{14} \oplus \underline{D'_{14:0}} = a_{14} \oplus' b_{14} \oplus \underline{D_{14:0}}$

where ⊕' denotes the Exclusive NOR operation. A method is now disclosed for constructing $D_{14:0}$ which is suitable for dynamic circuit techniques.

$$D_{14:0} = D_{14:9}[B_{14:12} + B_{11:9} + D_{8:7}[B_{8:7} + G_{6:5}] + P_{8:6}P_{5:5}D_{4:3}[B_{4:3} + D_{2:0}]]$$

The building blocks for the bracketed terms are:

$$B_{4:3} + D_{2:0} = g_4 + g_3 + g_2 + p_2g_1 + p_2p_1p_0$$

$$P_{5:5}\,D_{4:3} = p_5g_4 + p_5p_4p_3$$

$$P_{8:6} = p_8p_7p_6$$

$$B_{8:7} + G_{6:5} = g_8 + g_7 + g_6 + p_6g_5$$

$$D_{8:7} = g_8 + p_{8p7}$$

$$B_{11:9} = g_{11} + g_{10} + g_9$$

$$B_{14:12} = g_{14} + g_{13} + g_{12}$$

$D_{14:9}$ is derived as follows:

$$D_{14:9} = D_{14:12}[B_{14:12} + D_{11:9}]$$

Having building blocks:

$$B_{14:12} = g_{14} + g_{13} + g_{12} + g_{11}$$

$$D_{11:9} = p_{11}g_{10} + p_{11}p_{10}p_9$$

$$D_{14:12} = g_{14} + p_{14}g_{13} + p_{14}p_{13}p_{12}$$

Thus far we have disclosed method and apparatus for a single carry generation logic unit. Given two n-bit binary numbers $a = a_{n-1}\ldots a_1a_0$ and $b = b_{n-1}\ldots b_1b_0$, their sum is the n+1 bit number given by $s = s_n\ldots s_1s_0$

$s_n = c_n$

$s_i = a_i \oplus b_i \oplus c_i$

where $c_i$ is the carry into position i. Thus it is required that all the carries be generated. It is required that this be done with the highest speed circuit together with efficient silicon utilization. This will now be illustrated by way of example. We consider a 27-bit adder.

$G_{26:0} = D_{26:14}K_{26:0}$

$K_{26:0} = B_{26:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{8:0} = B_{8:6} + K_{5:3} + P_{4:2}K_{2:0}$

$K_{17:9} = B_{17:15} + K_{14:12} + P_{13:11}K_{11:9}$

$K_{26:18} = B_{26:24} + K_{23:21} + P_{22:20}K_{20:18}$

$B_{26:18} = B_{26:24} + B_{23:21} + B_{20:18}$

$K_{2:0} = g_2 + g_1 + p_1g_0$

$K_{5:3} = g_5 + g_4 + p_4g_3$

$K_{8:6} = g_8 + g_7 + p_7g_6$

$K_{11:9} = g_{11} + g_{10} + p_{10}g_9$

$K_{14:12} = g_{14} + g_{13} + p_{13}g_{12}$

$K_{17:15} = g_{17} + g_{16} + p_{16}g_{15}$

$K_{20:18} = g_{20} + g_{19} + p_{19}g_{18}$

$K_{23:21} = g_{23} + g_{22} + p_{22}g_{21}$

$K_{26:24} = g_{26} + g_{25} + p_{25}g_{24}$

$B_{8:6} = g_8 + g_7 + g_6$

$B_{17:15} = g_{17} + g_{16} + g_{15}$

$B_{20:18} = g_{20} + g_{19} + g_{18}$

$B_{23:21} = g_{23} + g_{22} + g_{21}$

$B_{26:24} = g_{26} + g_{25} + g_{24}$

$D_{26:14} = D_{26:23}K_{26:18} + P_{26:18}D_{17:14}$

$D_{26:23} = p_{26}K_{26:24} + p_{26}P_{25:23}$

$D_{17:14} = p_{17}K_{17:15} + p_{17}P_{16:14}$

$P_{13:9}D_{8:5} = [P_{13:11}P_{10:8}][K_{8:6} + P_{7:5}]$

$P_{26:18} = P_{26:24}P_{23:21}P_{20:18}$

$P_{4:2} = p_4p_3p_2$

$P_{7:5} = p_7p_6p_5$

$P_{10:8} = p_{10}p_9p_8$

$P_{13:11} = p_{13}p_{12}p_{11}$

$P_{20:18} = p_{20}p_{19}p_{18}$

$P_{22:20} = p_{22}p_{21}p_{20}$

$P_{23:21} = p_{23}p_{22}p_{21}$

$P_{26:24} = p_{26}p_{25}p_{24}$

This completes the circuit for $G_{26:0}$. We now present the remaining $G_{i:0}$.

$G_{25:0} = D_{25:14}K_{25:0}$

$K_{25:0} = B_{25:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$B_{25:18} = B_{25:24} + B_{23:21} + B_{20:18}$

$B_{25:24} = g_{25} + g_{24}$

$D_{25:14} = D_{25:23}K_{25:18} + P_{25:18}D_{17:14}$

$K_{25:18} = B_{25:24} + K_{23:21} + P_{22:20}K_{20:18}$

$D_{26:23} = p_{25}B_{25:24} + P_{25:23}$

$P_{25:18} = P_{25:24}P_{23:21}P_{20:18}$

$P_{25:23} = p_{25}p_{24}p_{23}$

$P_{25:24} = p_{25}p_{24}\ P_{4:2} = p_4p_3p_2$

$G_{24:0} = D_{24:14}K_{24:0}$

$K_{24:0} = K_{24:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{24:18} = g_{24} + K_{23:21} + P_{22:20}K_{20:18}$

$D_{24:14} = D_{24:23}K_{24:18} + P_{24:18}D_{17:14}$

$D_{24:23} = g_{24} + P_{24:23}$

$P_{24:18} = p_{24}P_{23:21}P_{20:18}$

$P_{24:23} = p_{24}p_{23}$

$G_{23:0} = D_{23:14}K_{23:0}$

$K_{23:0} = K_{23:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{23:18} = K_{23:21} + P_{22:20}K_{20:18}$

$D_{23:14} = p_{23}K_{23:18} + P_{23:18}D_{17:14}$

$P_{23:18} = P_{23:21}P_{20:18}$

$G_{22:0} = D_{22:14}K_{22:0}$

$K_{22:0} = K_{22:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{22:18} = K_{22:21} + P_{22:20}K_{20:18}$

$K_{22:21} = g_{22} + g_{21}$

$D_{22:14} = p_{22}K_{22:18} + P_{22:18}D_{17:14}$

$P_{22:18} = P_{22:21}P_{20:18}$

$P_{22:21} = p_{22}p_{21}$

$G_{21:0} = D_{21:14}K_{21:0}$

$K_{21:0} = K_{21:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{21:18} = g_{21} + P_{21:20}K_{20:18}$

$D_{21:14} = p_{21}K_{21:18} + P_{21:18}D_{17:14}$

$P_{21:18} = p_{21}P_{20:18}$

$G_{20:0} = D_{20:14}K_{20:0}$

$K_{20:0} = K_{20:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$D_{20:14} = p_{20}K_{20:18} + P_{20:18}D_{17:14}$

$G_{19:0} = D_{19:14}K_{19:0}$

$K_{19:0} = K_{19:18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{19:18} = g_{19} + g_{18}$

$D_{19:14} = p_{19}K_{19:18} + P_{19:18}D_{17:14}$

$P_{19:18} = p_{19}p_{18}$

$G_{18:0} = D_{18:14}K_{18:0}$

$K_{18:0} = g_{18} + K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$D_{18:14} = g_{18} + p_{18}D_{17:14}$

$G_{17:0} = D_{17:14}K_{17:0}$

$K_{17:0} = K_{17:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$G_{16:0} = D_{16:14}K_{16:0}$

$K_{16:0} = K_{16:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{16:9} = K_{16:15} + K_{14:12} + P_{13:11}K_{11:9}$

$K_{16:15} = g_{16} + g_{15}$

$D_{16:14} = p_{16}K_{16:15} + P_{16:14}$

$P_{16:14} = p_{16}p_{15}p_{14}$

$G_{15:0} = D_{15:14}K_{15:0}$

$K_{15:0} = K_{15:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{15:9} = g_{15} + K_{14:12} + P_{13:11}K_{11:9}$

$D_{15:14} = g_{15} + p_{15}p_{14}$

$G_{14:0} = p_{14}K_{14:0}$

$K_{14:0} = K_{14:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{14:9} = K_{14:12} + P_{13:11}K_{11:9}$

$G_{13:0} = p_{13}K_{13:0}$

$K_{13:0} = K_{13:9} + [P_{13:9}D_{8:5}]K_{8:0}$

$K_{13:9} = K_{13:12} + P_{13:11}K_{11:9}$

$K_{16:12} = g_{13} + g_{12}$

$G_{12:0} = p_{13}K_{12:0}$

$K_{12:0} = K_{12:9} + [P_{12:9}D_{8:5}]K_{8:0}$

$P_{12:9}D_{8:5} = [P_{12:11}P_{10:8}][\ K_{8:6} + P_{7:5}]$

$K_{12:9} = g_{12} + P_{12:11}K_{11:9}$

$P_{12:11} = p_{12}p_{11}$

$G_{11:0} = p_{11}K_{11:0}$

$K_{11:0} = K_{11:9} + [P_{11:9}D_{8:5}]K_{8:0}$

$P_{11:9}D_{8:5} = [p_{11}P_{10:8}][\ K_{8:6} + P_{7:5}]$

$K_{11:9} = p_{11}K_{11:9}$

$G_{10:0} = p_{10}K_{10:0}$

$K_{10:0} = K_{10:9} + [P_{10:9}D_{8:5}]K_{8:0}$

$P_{10:9}D_{8:5} = P_{10:8}[\ K_{8:6} + P_{7:5}]$

$K_{10:9} = g_{10} + g_9$

$G_{9:0} = p_9K_{9:0}$

$K_{9:0} = g_9 + [p_9D_{8:5}]K_{8:0}$

$p_9D_{8:5} = P_{9:8}[\ K_{8:6} + P_{7:5}]$

$P_{9:8} = p_9p_8$

$G_{8:0} = D_{8:5}K_{8:0}$

$D_{8:5} = p_8K_{8:6} + p_8P_{7:5}$

$G_{7:0} = D_{7:5}K_{7:0}$

$K_{7:0} = K_{7:6} + K_{5:3} + P_{4:2}K_{2:0}$

$D_{7:5} = K_{7:6} + P_{7:5}$

$K_{7:6} = g_7 + g_6$

$G_{6:0} = D_{6:5}K_{6:0}$

$K_{6:0} = g_6 + K_{5:3} + P_{4:2}K_{2:0}$

$D_{6:5} = g_6 + P_{6:5}$

$P_{6:5} = p_6p_5$

$G_{5:0} = p_5K_{5:0}$

$K_{5:0} = K_{5:3} + P_{4:2}K_{2:0}$

$G_{4:0} = G_{4:3} + P_{4:2}K_{2:0}$

$G_{4:3} = g_4 + p_4g_3$

$G_{3:0} = g_3 + P_{3:2}K_{2:0}$

$P_{3:2} = p_3p_2$

$G_{2:0} = p_2K_{2:0}$

$G_{1:0} = g_1 + p_1g_0$

$G_{0:0} = g_0$

The present invention is not limited to use for addition and subtraction, but may also have other applications. For example, two numbers may be compared by generating the most signifcant carry bit for the difference between the two numbers. It may not be necessary to generate the other carry bits of this subtraction, or to actually perform the subtraction. It may also not be necessary to take all of the least significant bits of the two numbers into account when performing a comparison – the number may, in effect, be rounded up or down before performing a comparison. Thus, it is not essential that a circuit according to the invention should input all of the least significant bits of the two input numbers, in order to generate a carry bit and perform a useful comparison of the numbers.

It is not essential that the two input numbers a and b have the same number of digits. If they do not, then either leading zeros may be added to the smaller number if necessary, or the hardware may be hardwired to set generate functions to zero for the most significant digits which are only present in one of the numbers, and set the

propagate functions in the corresponding column to equal the value of the other input number bits.

The above generally describes a logic circuit for generation of a carry or sum bit output by combining two sets of binary inputs, the logic circuit comprises bit level carry generate and propagate function logic for receiving the binary inputs and for generating bit level carry generate and propagate function bits for said binary inputs by respectively logically AND and OR combining respective bits of said binary inputs; first logic for receiving bit level carry generate and propagate function bits for a first group of at least three most significant bits of said binary inputs to generate a high output if a carry is generated out of the first group of most significant bits of said binary input or if said carry propagate function bits for the most significant bits are all high; second logic for receiving bit level carry generate and propagate function bits for said binary inputs to generate a high output if any of said carry generate function bits for the most significant bits are high or if a carry is generated out of a second group of least significant bits of said binary input; and combining logic for generating the carry or sum bit output by combining outputs of said first and second logic.

Although the present invention has been described with reference to specific embodiments, it will be apparent to a skilled person in the art that modifications lie within the spirit and scope of the present invention. Any documents referred to above are hereby incorporated by reference for any purpose.